

# 3 speed Rotary Encoder Full Step library for Arduino

---

This is an optimized three speed Rotary Encoder library for Arduino which supports:

- Full step Rotary Encoder types.
- Detect three rotation speeds.
- Configurable rotation speed sensitivity.
- Polling and interrupt based.
- Single or multiple Rotary Encoders.
- Optional Rotary button.
- Pin state table in flash.

## Full step / half step Rotary Encoders

---

The difference between a full step or half step Rotary Encoder type is how the data signals of the two pins are generated. It depends on the mechanical construction of the notches and contacts inside the Rotary Encoder.

Please refer to the [ErriezRotaryEncoderHalfStep](#) library for half step Rotary Encoders. Experiment with the half step and full step libraries which works optimal for your Rotary Encoder.

## Hardware

---

Connect the two rotary pins to the DIGITAL pins of an Arduino board.

A third rotary button pin is not used in the Rotary library, but can be used in the sketch.

Tested with Arduino IDE v1.8.5 on hardware:

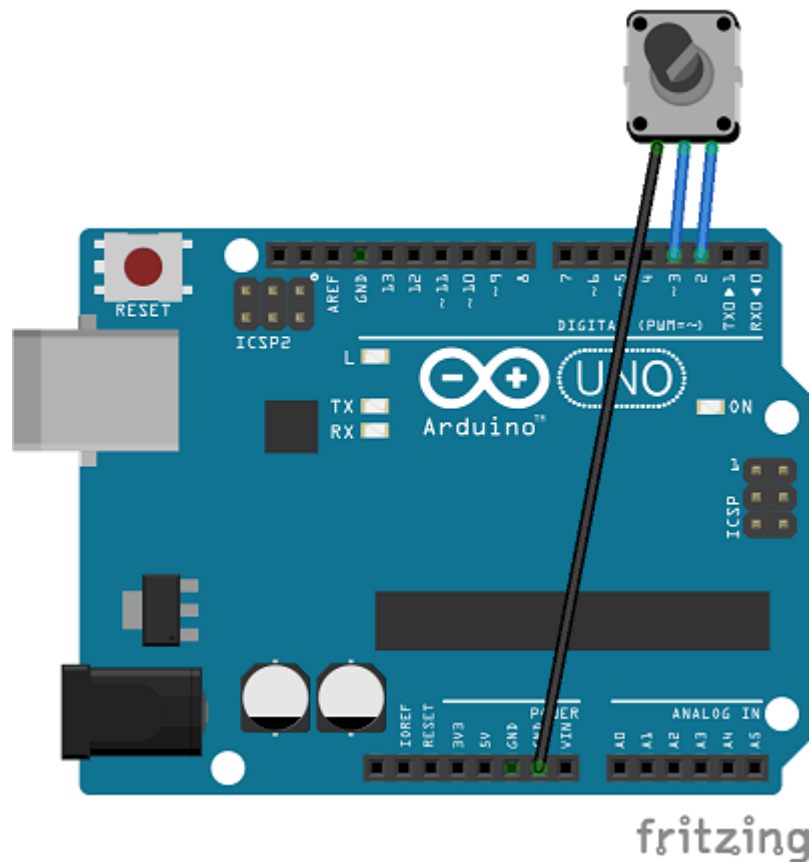
- Arduino UNO
- Arduino Nano
- Arduino Micro
- Arduino Pro or Pro Mini
- Arduino Mega or Mega2560
- Arduino Leonardo
- WeMos D1 R2 & mini (ESP8266)

## Interrupts

Both rotary pins must be connected to a DIGITAL pin with interrupt support, such as `INT0` or `INT1`. This is chip specific. Please refer to the documentation of your board or [attachInterrupt\(\)](#).

## Arduino UNO hardware

The connection below can be used for polled and interrupts. An optional button pin can be connected to DIGITAL pin 4.



Rotary pin	Arduino UNO/NANO/Mega2560/Leonardo board
1	D2 (INT0)
2	D3 (INT1)
Button (optional)	D4
GND	GND

## Arduino WeMos D1 R2 & mini (ESP8266) hardware

Note that some ESP8266 pins mixes ESP8622 GPIO pins with Arduino digital pins. Connect a Rotary Encoder to the following pins which can be used with polled and interrupt examples:

Rotary pin	ESP8622 pin	Text on board WeMos D1 R2
1	GPIO13	D7 MOSI
2	GPIO12	D6 MISO
Button (optional)	GPIO14	D5 SCK
LED (Not used)	GPIO2	D4
GND	GND	GND

**Note:** An external pull-up resistor is required when a pin does not have an internal pull-up.

```

1 // Connect the rotary pins to the WeMos D1 R2 board:
2 #define ROTARY_PIN1      12
3 #define ROTARY_PIN2      13
4 #define ROTARY_BUTTON_PIN 14

```

## Examples

The following examples are available:

- Rotary | Interrupt | [InterruptFullStepBasic](#)
- Rotary | Interrupt | [InterruptFullStepButton](#)
- Rotary | Interrupt | [InterruptFullStepCounter](#)
- Rotary | Polled | [PolledFullStepBasic](#)
- Rotary | Polled | [PolledFullStepButton](#)
- Rotary | Polled | [PolledFullStepCounter](#)
- Rotary | Polled | [PolledFullStepMultiple](#)

## Usage

### Read rotary with polling

```

1 #include <RotaryFullStep.h>
2
3 // Connect rotary pins to the DIGITAL pins of the Arduino board
4 #define ROTARY_PIN1  2
5 #define ROTARY_PIN2  3
6
7 // Enable ONE of the three constructors below with different number of arguments:
8
9 // Initialize full step rotary encoder, default pull-up enabled, default
10 // sensitive=100

```

```

11 RotaryFullStep rotary(ROTARY_PIN1, ROTARY_PIN2);
12
13 // Or initialize full step rotary encoder, pull-up disabled, default sensitive=100
14 // RotaryFullStep rotary(ROTARY_PIN1, ROTARY_PIN2, false);
15
16 // Or initialize full step rotary encoder, pull-up enabled, sensitive 1..255
17 // A higher value is more sensitive
18 // RotaryFullStep rotary(ROTARY_PIN1, ROTARY_PIN2, true, 150);
19
20 void loop()
21 {
22     int rotaryState = rotary.read();
23
24     // rotaryState = -3: Counter clockwise turn, multiple notches fast
25     // rotaryState = -2: Counter clockwise turn, multiple notches
26     // rotaryState = -1: Counter clockwise turn, single notch
27     // rotaryState = 0: No change
28     // rotaryState = 1: Clockwise turn, single notch
29     // rotaryState = 2: Clockwise turn, multiple notches
30     // rotaryState = 3: Clockwise turn, multiple notches fast
31 }

```

## Read rotary with interrupts

```

1  #include <RotaryFullStep.h>
2
3  // Connect rotary pins to Arduino DIGITAL pins with interrupt support:
4  //
5  // +-----+-----+
6  // |           Board           | DIGITAL interrupt pins |
7  // +-----+-----+
8  // | Uno, Nano, Mini, other 328-based | 2, 3             |
9  // | Mega, Mega2560, MegaADK         | 2, 3, 18, 19, 20, 21 |
10 // | Micro, Leonardo, other 32u4-based | 0, 1, 2, 3, 7         |
11 // +-----+-----+
12 //
13 #define ROTARY_PIN1  2
14 #define ROTARY_PIN2  3
15
16 // Enable ONE of the three constructors below with different number of arguments:
17
18 // Initialize full step rotary encoder, default pull-up enabled, default
19 // sensitive=100
20 RotaryFullStep rotary(ROTARY_PIN1, ROTARY_PIN2);
21
22 // Or initialize full step rotary encoder, pull-up disabled, default sensitive=100
23 // RotaryFullStep rotary(ROTARY_PIN1, ROTARY_PIN2, false);
24
25 // Or initialize full step rotary encoder, pull-up enabled, sensitive 1..255
26 // A higher value is more sensitive
27 // RotaryFullStep rotary(ROTARY_PIN1, ROTARY_PIN2, true, 150);
28
29 void setup()

```

```

30 {
31     // Initialize pin change interrupt on both rotary encoder pins
32     attachInterrupt(digitalPinToInterrupt(ROTARY_PIN1), rotaryInterrupt, CHANGE);
33     attachInterrupt(digitalPinToInterrupt(ROTARY_PIN2), rotaryInterrupt, CHANGE);
34 }
35
36 void rotaryInterrupt()
37 {
38     int rotaryState = rotary.read();
39
40     // rotaryState = -3: Counter clockwise turn, multiple notches fast
41     // rotaryState = -2: Counter clockwise turn, multiple notches
42     // rotaryState = -1: Counter clockwise turn, single notch
43     // rotaryState = 0: No change
44     // rotaryState = 1: Clockwise turn, single notch
45     // rotaryState = 2: Clockwise turn, multiple notches
46     // rotaryState = 3: Clockwise turn, multiple notches fast
47 }

```

## Library installation using Git

`Git` is the preferred way to keep this library up to date, because the Arduino Library manager does not update as long as this library is not added to the official Arduino Library database.

### Install Git client for Windows

Install a [Git client for Windows](#).

### Install Git client for Linux

Open a command prompt and install a Git client for Linux, such as Debian Ubuntu:

```
1 | sudo apt-get install git
```

### Get Arduino libraries directory

This library must be installed in the Arduino Sketchbook library subdirectory.

To retrieve the Arduino Sketchbook directory, open the Arduino IDE Preferences dialog box via: `File` | `Preferences` | `Settings tab` and copy the Sketchbook location.

For example on:

- Windows : `C:\Users\User\Documents\Arduino`
- Linux: `/home/user/Arduino`

### Clone this library

Clone this library by opening a command prompt:

- Windows: ( `Windows key + R`, Type `cmd` + `[ENTER]` )

- Linux: Depends on your version.

Then type:

```
1 # Change directory to the sketchbook directory as configured in the Arduino IDE:
2 # Windows:
3 cd C:\Users\User\Documents\Arduino
4 # Linux:
5 cd ~/Arduino
6
7 # Go to the libraries subdirectory
8 cd libraries
9
10 # Run the git clone library once:
11 git clone https://github.com/Erriez/ErriezRotaryEncoderFullStep.git
```

**IMPORTANT:** Restart the Arduino IDE.

## Update this library

Open a command prompt and type:

```
1 # Change directory to the sketchbook directory as configured in the Arduino IDE:
2 # Windows:
3 cd C:\Users\User\Documents\Arduino
4 # Linux:
5 cd ~/Arduino
6
7 # Go to the libraries subdirectory
8 cd libraries
9
10 # Update the library:
11 git pull
```

**IMPORTANT:** Restart the Arduino IDE.

## Library installation with a ZIP

---

This method is not preferred, because updates should be manually installed.

1. Download [the latest version here](#).
2. Open the Arduino IDE.
3. Sketch | Include Library | Add .ZIP library...
4. Browse to the downloaded ZIP.
5. Restart the IDE.